In re Patent Application of:
SANCHES ET AL.
Serial No. **09/915,761**
Filing Date: **JULY 26, 2001**

In the Claims:

Claims 1 to 19 (Cancelled).

20. (Currently Amended) A signal processor for executing variable-sized instructions, each instruction comprising up to N ~~codes,~~ codes with N being a positive integer greater than 1, the signal processor comprising:

a program memory comprising I individually addressable, parallel-connected memory banks with I being a positive integer at least equal to N, said program memory comprising a program recorded in an interlaced fashion as a function ~~at a rate~~ of one code per memory bank and per address applied to said memory banks; and

reading means for reading said program memory by reading a code in each of said I memory banks during a cycle for reading an instruction, with each instruction comprising a sequence of codes to be read ~~with a cycle comprising at least one code to be read,~~ and when a number of the sequence of codes of the instruction being read is less than I, then ~~comprises~~ codes belonging to a following ~~instruction.~~ instruction are read, said reading means comprising

address means for applying to said memory banks individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the cycle for reading the instruction, by a value equal to a number of codes belonging to a previous instruction, and applying to each of said memory banks an individual read address that is based upon a result of a division by I of the collective value of the program counter, and

filtering means for filtering codes that do not belong to the instruction to be read, while using parallelism bits accompanying the codes.

Claim 21 (Cancelled).

22. (Currently Amended) A signal processor according to ~~Claim 21,~~ Claim 20, wherein ~~said address means applies to each of said memory banks an~~ the individual read address for each respective memory bank is equal to P0 or P0+1, with P0 being a quotient of ~~a~~ the division by I of ~~a~~ the collective value of the program counter.

23. (Currently Amended) A signal processor according to Claim 22, wherein said address means comprises applying, to an ~~i~~ ix ranking memory bank, an address equal to P0 when ~~i~~ ix is greater than R and an address equal to P0+1 when ~~i~~ ix is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

24. (Currently Amended) A signal processor according to Claim 20, wherein said reading means comprises reorganization means for reorganizing codes of a sequence of codes read in said program memory according to an algorithm defined as follows:

c'(j) = ~~c(i),~~ c(ix), ~~and~~ with ~~i~~ ix = (j+R')modulo I,

and with ~~i~~ ix and j designating ~~ranks~~ a ranking of the codes before and after reorganization, ~~c(i)~~ c(ix) designating ~~i~~ ix as the ranking of the codes before reorganization, ~~in their arrangement after being read in said program memory,~~ c'(j) designating j as the ranking of the codes after reorganization, and R' is a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

25. (Previously Presented) A signal processor according to Claim 24, wherein said reorganization means applies to the codes of the sequence of codes read a circular permutation comprising a number of circular permutations equal to R' or to I-R', depending on a direction of the circular permutation made.

26. (Previously Presented) A signal processor according to Claim 25, wherein said reorganization means comprises a barrel shifter having a control input for receiving the parameter R'.

Claim 27 (Cancelled).

28. (Currently Amended) A signal processor according to ~~Claim 27,~~ Claim 20, wherein the filtered codes are replaced by no-operation codes.

29. (Previously Presented) A signal processor according to Claim 28, wherein said filtering means executes an algorithm defined as follows:

For j = 0,

val(j=0) = v,

s(j=0) = c'(j=0);

For j going from 1 to I,

val(j) = v if:

val(j-1) = v and if parallelism bit of c'(j)= p,

else val(j-1) = /v;

s(j) = c'(j) if val(j) = v;

s(j) = NOP if val(j) = /v,

with val(j) being a validation term associated with each j ranking code, c'(j) is capable of having two values v and /v, s(j) designates j ranking outputs of said filtering means corresponding to same ranking inputs receiving a code c'(j), and NOP indicates a no-operation code.

30. (Previously Presented) A signal processor according to Claim 29, wherein said reading means comprises at least one parallel-connected RISC type execution unit for receiving non-filtered codes.

31. (Currently Amended) A processor for executing variable-sized instructions, each instruction comprising up to N ~~codes,~~ codes with N being a positive integer greater than 1, the processor comprising:

a memory comprising I individually addressable, parallel-connected memory banks with I being a positive integer at least equal to N, said memory comprising a program recorded in an interlaced fashion; and

a reading circuit for reading said memory by reading a code in each of said I memory banks during a cycle for reading an instruction, with each instruction comprising a sequence of codes to be read ~~with a cycle comprising at least one code to be read,~~ and when a number of the sequence of codes of the instruction being read is less than I, then ~~comprises~~ codes belonging to a following ~~instruction.~~ instruction are read, said reading circuit comprising

an address circuit for applying to said memory banks individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the cycle for reading the instruction, by a value equal to a number of codes belonging to a previous instruction, and applying to each of said memory banks an individual read address that is based upon a result of a division by I of the collective value of the program counter, and

a filtering circuit for filtering codes that do not belong to the instruction to be read, while using parallelism bits accompanying the codes.

32. (Currently Amended) A processor according to Claim 31, wherein the program is recorded as a function ~~at a rate~~ of one code per memory bank and per address applied to said memory banks.

Claim 33 (Cancelled).

34. (Currently Amended) A processor according to ~~Claim 33~~ Claim 31, wherein ~~said address circuit applies to each of said memory banks an~~ the individual read address for each respective memory bank is equal to P0 or P0+1, with P0

being a quotient of ~~a~~ the division by I of ~~a~~ the collective value of the program counter.

35. (Currently Amended) A processor according to Claim 34, wherein said address circuit comprises applying, to an ~~i~~ ix ranking memory bank, an address equal to P0 when ~~i~~ ix is greater than R and an address equal to P0+1 when ~~i~~ ix is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

36. (Currently Amended) A processor according to Claim 31, wherein said reading circuit comprises a reorganization circuit for reorganizing codes of a sequence of codes read in said memory according to an algorithm defined as follows:

c'(j) = ~~c(i),~~ c(ix), ~~and~~ with ~~i~~ ix = (j+R')modulo I,

and with ~~i~~ ix and j designating ~~ranks~~ a ranking of the codes before and after reorganization, ~~c(i)~~ c(ix) designating ~~i~~ ix as the ranking of the codes before reorganization, ~~in their arrangement after being read in said program memory,~~ c'(j) designating j as the ranking of the codes after reorganization, and R' is a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

Claim 37 (Cancelled).

38. (Currently Amended) A processor according to

~~Claim 37,~~ Claim 31, wherein the filtered codes are replaced by no-operation codes.

39. (Currently Amended) A processor according to Claim 38, wherein said filtering circuit executes an algorithm defined as follows:

For j = 0,
val(j=0) = v,
s(j=0) = c'(j=0);
For j going from 1 to I,
val(j) = v if:
val(j-1) = v and if parallelism bit of c'(j)= p,
else val(j-1) = /v;
s(j) = c'(j) if val(j) = v;
s(j) = NOP if val(j) = /v,

with val(j) being a validation term associated with each j ranking code, c'(j) is capable of having two values v and /v, s(j) designates j ranking outputs of said filtering ~~means~~ circuit corresponding to same ranking inputs receiving a code c'(j), and NOP indicates a no-operation code.

40. (Currently Amended) A processor according to Claim 39, wherein said reading ~~means~~ circuit comprises at least one parallel-connected RISC type execution unit for receiving non-filtered codes.

41. (Currently Amended) A method for reading variable-sized instructions in a signal processor, with each instruction comprising up to N ~~codes,~~ codes with N being a positive integer greater than 1, the method comprising:

providing a program memory comprising I individually addressable, parallel-connected memory banks with I being a positive integer at least equal to N;

recording codes of a program in the program memory in an interlaced fashion as a function ~~at a rate~~ of one code per bank and per address applied to the memory banks; ~~and~~

applying, to the memory banks, individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the read cycle for the instruction, by a value equal to a number of codes contained in a previous instruction, and applying to each of the memory banks an individual read address that is based upon a result of a division by I of the collective value of the program counter; and

during a read cycle of an instruction, with each instruction comprising a sequence of codes to be read, reading ~~a~~ the sequence of codes ~~in the I memory banks, the sequence comprising at least one code of the instruction to be read and,~~ and when a number of the sequence of codes ~~of the instruction~~ read is less than I, ~~comprises~~ then reading codes belonging to a following ~~instruction.~~ instruction; and

filtering codes read that do not belong to the instruction, while using parallelism bits accompanying the codes.

Claim 42 (Cancelled).

43. (Currently Amended) A method according to ~~Claim 42,~~ Claim 41, ~~further comprising applying, to each to the memory banks, an~~ wherein the individual read address for each

respective memory bank is equal to P0 or P0+1, with P0 being a quotient of ~~a~~ the division by I of the collective value of the program counter.

44. (Currently Amended) A method according to Claim 43, further comprising applying, to an ~~i~~ ix ranking memory bank, an address equal to P0 when ~~i~~ ix is greater than R and an address equal to P0+1 when ~~i~~ ix is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

45. (Currently Amended) A method according to Claim 41, further comprising reorganizing codes of the sequence of codes read in the program memory according to an algorithm defined as follows:

c'(j) = ~~c(i),~~ c(ix), ~~and~~ with ~~i~~ ix = (j+R')modulo I,

and with ~~i~~ ix and j designating ~~ranks~~ a ranking of the codes before and after reorganization, ~~c(i)~~ c(ix) designating ~~i~~ ix as the ranking of the codes before reorganization, ~~in their arrangement after being read in the program memory,~~ c'(j) designating j as the ranking of the codes after reorganization, and R' is a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

Claim 46 (Cancelled).

47. (Currently Amended) A method according to ~~Claim~~

~~46,~~ Claim 41, wherein the filtered codes are replaced by no-operation codes.

48. (Previously Presented) A method according to Claim 47, wherein the codes are filtered according to an algorithm defined as follows:

For j = 0,
val(j=0) = v,
s(j=0) = c'(j=0);
For j going from 1 to I,
val(j) = v if:
val(j-1) = v and if parallelism bit of c'(j)= p,
else val(j-1) = /v;
s(j) = c'(j) if val(j) = v;
s(j) = NOP if val(j) = /v,

with val(j) being a validation term associated with each j ranking code, c'(j) is capable of having two values v and /v, s(j) designates j ranking outputs of the filtering corresponding to same ranking inputs receiving a code c′(j), and NOP indicates a no-operation code.

49. (Previously Presented) A method according to Claim 48, wherein non-filtered codes are sent to parallel-connected RISC type execution units.

50. (Currently Amended) A method for reading variable-sized instructions in a processor, with each instruction comprising up to N ~~codes,~~ codes with N being a positive integer greater than 1, the ~~signal~~ processor comprising a memory comprising I individually addressable,

parallel-connected memory banks, with I being a positive integer at least equal to N, the method comprising:

recording codes of a program in the memory in an interlaced fashion; ~~and~~

applying, to the memory banks, individual addresses generated from a collective value of a program counter that is incremented, before a beginning of the read cycle for the instruction, by a value equal to a number of codes contained in a previous instruction, and applying to each of the memory banks an individual read address that is based upon a result of a division by I of the collective value of the program counter; and

during a read cycle of an instruction, with each instruction comprising a sequence of codes to be read, reading ~~a~~ the sequence of codes ~~in the I memory banks, the sequence comprising at least one code of the instruction to be read and,~~ and when a number of the sequence of codes ~~of the instruction~~ read is less than I, ~~comprises~~ then reading codes belonging to a following ~~instruction.~~ instruction; and

filtering codes read that do not belong to the instruction, while using parallelism bits accompanying the codes.

51. (Currently Amended) A method according to Claim 50, wherein the program is recorded as a function ~~at a rate~~ of one code per bank and per address applied to the memory banks

Claim 52 (Cancelled).

53. (Currently Amended) A method according to ~~Claim~~

~~52,~~ Claim 50, ~~further comprising applying, to each to the memory banks, an~~ wherein the individual read address for each respective memory bank is equal to P0 or P0+1, with P0 being a quotient of ~~a~~ the division by I of the collective value of the program counter.

54. (Currently Amended) A method according to Claim 53, further comprising applying, to an ~~i~~ ix ranking memory bank, an address equal to P0 when ~~i~~ ix is greater than R and an address equal to P0+1 when ~~i~~ ix is less than or equal to R, with R being a remainder of the division by I of the value of the program counter.

55. (Currently Amended) A method according to Claim 50, further comprising reorganizing codes of the sequence of codes read in the memory according to an algorithm defined as follows:

c'(j) = ~~c(i),~~ c(ix), ~~and~~ with ~~i~~ ix = (j+R')modulo I,

and with ~~i~~ ix and j designating ~~ranks~~ a ranking of the codes before and after reorganization, ~~c(i)~~ c(ix) designating ~~i~~ ix as the ranking of the codes before reorganization, ~~in their arrangement after being read in the program memory,~~ c'(j) designating j as the ranking of the codes after reorganization, and R' is a remainder of a division by I of a value that was shown by the program counter during a previous clock cycle.

Claim 56 (Cancelled).

57. (Currently Amended) A method according to ~~Claim 56,~~ Claim 50, wherein the filtered codes are replaced by no-operation codes.

58. (Previously Presented) A method according to Claim 57, wherein the codes are filtered according to an algorithm defined as follows:

For j = 0,
val(j=0) = v,
s(j=0) = c'(j=0);
For j going from 1 to I,
val(j) = v if:
val(j-1) = v and if parallelism bit of c'(j)= p,
else val(j-1) = /v;
s(j) = c'(j) if val(j) = v;
s(j) = NOP if val(j) = /v,

with val(j) being a validation term associated with each j ranking code, c'(j) is capable of having two values v and /v, s(j) designates j ranking outputs of the filtering corresponding to same ranking inputs receiving a code c'(j), and NOP indicates a no-operation code.

59. (Previously Presented) A method according to Claim 58, wherein non-filtered codes are sent to parallel-connected RISC type execution units.